Clustering Stream and/or Instruction Queues for Multi-Streaming Processors

by inventors
Mario Nemirovsky, Stephen W. Melvin and Nandakumar Sampath,
Enric Musoll, and Hector Urdaneta

Field of the Invention

The present invention is in the field of digital processing and pertains more particularly to architecture and operation in dynamic multistreaming processors.

Background of the Invention

Conventional pipelined single-stream processors incorporate fetch and dispatch pipeline stages, as is true of most conventional processors. In such processors, in the fetch stage, one or more instructions are read from an instruction cache and in the dispatch stage, one or more instructions are sent to execution units (EUs) to execute. The stages may be separated by one or more other stages, for example a decode stage. In such a processor the fetch and dispatch stages are coupled together such that the fetch stage generally fetches from the instruction stream in every cycle.

In multistreaming processors known to the present inventors, multiple instruction streams are provided, each having access to the execution units. Multiple fetch stages may be provided, one for each instruction stream, although one dispatch stage is employed. Thus, the fetch and dispatch stages are coupled to one another as in other conventional processors, and each instruction stream generally fetches instructions in each cycle. That is, if there are five instruction streams, each of the five fetches in each cycle, and there needs to be a port to the instruction cache for each stream, or a separate cache for each stream.

In a multistreaming processor multiple instruction streams share a common set of resources, for example execution units and/or access to memory resources. In such a processor, for example, there may be M instruction streams that share Q execution units in any given cycle. This means that a set of up to Q instructions is chosen from the M instruction streams to be delivered to the execution units in each cycle. In the following cycle a different set of up to Q instructions is chosen, and so forth. More than one instruction may be chosen from the same instruction stream, up to a maximum P, given that there are no dependencies between the instructions.

It is desirable in multistreaming processors to maximize the number of instructions executed in each cycle. This means that the set of up to Q instructions that is chosen in each cycle should be as close to Q as possible. Reasons that there may not be Q instructions available include flow dependencies, stalls due to memory operations, stalls due to branches, and instruction fetch latency.

A further difficulty in multi-streaming processors, particularly is such processors having a relatively large number of streams, is in operating the processor over all of the streams.

What is clearly needed in the art is an apparatus and method to cluster streams in a multi-streaming processor, such that separate clusters can operate substantially independently. The present invention, in several embodiments described in enabling detail below, provides a unique solution.

Summary of the Invention

In a preferred embodiment of the present invention a pipelined multistreaming processor is provided, comprising an instruction source, a first cluster of a plurality of streams fetching instructions from the instruction source, a second cluster of a plurality of streams fetching instructions from the instruction source, dedicated instruction queues for individual streams in each cluster, a first dedicated dispatch stage in the first cluster for dispatching instructions to execution units, and a second dedicated dispatch stage in the second cluster for selecting and dispatching instructions to execution units. The processor is characterized in that the clusters operate independently, with the dedicated dispatch stage taking instructions only from the instruction queues in the individual clusters to which the dispatch stages are dedicated.

In some embodiments individual ones or groups of execution units are associated with and dedicated for use by individual clusters. Also in some embodiments individual streams in a cluster have one or both of dedicated fetch and dispatch stages. In a particular embodiment the total number of streams in the processor is eight, with four streams in each cluster, and one stream from each cluster fetches instructions from the instruction source in each cycle. Further, in the particular embodiment, the select system may monitor a set of fetch program counters (FPC) having one FPC dedicated to each stream, and direct fetching if instructions beginning at addresses according to the to the program counters. Still further, in a particular embodiment, each stream selected to fetch for a cluster is directed to fetch eight instructions from the instruction source.

In some cases there may be one or more general execution units to which either or both dispatch stages may dispatch instructions. Also in preferred embodiments, each stream in each cluster has an associated instruction queue.

In another aspect of the invention, in a pipelined multistreaming processor having an instruction source and a plurality of streams, a method for simplifying implementation and operation of the streams is provided, comprising the steps of (a) clustering the streams into two or more clusters, with each cluster having a fetch stage; (b) dedicating a dispatch stage to each cluster, for dispatching instructions to execution units; and (c) fetching, in each cycle, a series of instructions from the instruction source by a single cluster.

In some embodiments of this method there groups of execution units dedicated to each cluster, to which the dispatch stages in that cluster may dispatch instructions. There are also, in some embodiments, one or both of fetch or dispatch stages dedicated to individual streams in a cluster. In a particular embodiment the total number of streams in the processor is eight, and the number of streams in each cluster is four. Also in a particular embodiment the select system monitors a set of fetch program counters (FPC) having one FPC associated with each stream, and directs fetching of instructions beginning at addresses according to the program counters. Further in a particular embodiment each stream selected to fetch is directed to fetch eight instructions from the instruction source.

In some embodiments of the method the processor further comprises one or more general execution units, and each dispatch stage is enabled to dispatch instructions to the general execution units. Also in some embodiments each stream in each cluster has an instruction queue associated with that stream, and further comprising a step for dispatching instructions to execution units dedicated to each cluster from the instruction queues associated with the streams in each cluster.

In embodiments of the present invention, described in enabling detail below, for the first time a pipelined, multi-streaming processor is provided, wherein streams may be clustered, and operations may therefore be more efficiently accomplished.

Brief Description of the Drawings

Fig. 1 is a block diagram depicting a pipelined structure for a processor in the prior art.

Fig. 2 is a block diagram depicting a pipelined structure for a multistreaming processor known to the present inventors.

Fig. 3 is a block diagram for a pipelined architecture for a multistreaming processor according to an embodiment of the present invention.

Fig. 4 is a block diagram for a pipelined architecture for a multistreaming processor according to another embodiment of the present invention.

Description of the Preferred Embodiments

Fig. 1 is a block diagram depicting a pipelined structure for a processor in the prior art. In this prior art structure there is an instruction cache 11, wherein instructions await selection for execution, a fetch stage 13 which selects and fetches instruction into the pipeline, and a dispatch stage which dispatches instructions to execution units (EUs) 17. In many conventional pipelined structures there are additional stages other than the exemplary stages illustrated here.

In the simple architecture illustrated in Fig. 1 everything works in lockstep. In each cycle an instruction is fetched, and another previously fetched instruction is dispatched to one of the execution units.

Fig. 2 is a block diagram depicting a pipelined structure for a multistreaming processor known to the present inventors, wherein a single instruction cache 19 has ports for three separate streams, and one instruction is fetched per cycle by each of three fetch stages 21, 23 and 25 (one for each stream). In this particular case a single dispatch stage 27 selects instructions from a pool fed by the three streams and dispatches those instructions to one or another of three execution units 29. In this architecture the fetch and dispatch units are still directly coupled. It should be noted that the architecture of Fig. 2, while prior to the present invention, is not necessarily in the public domain, as it is an as-yet proprietary architecture known to the present inventors. In another example, there may be separate caches for separate streams, but this does not provide the desired de-coupling.

Fig. 3 is a block diagram depicting an architecture for a dynamic multistreaming (DMS) processor according to an embodiment of the present invention. In this DMS processor there are eight streams and ten functional units, which may also be called execution units. Instruction cache 31 in this embodiment has two ports for providing instructions to fetch stage 33. Eight instructions may be fetched each cycle for each port, so 16 instructions may be fetched per cycle. The fetch stage is not explicitly shown in the staged pipeline as per the previous examples, but is described further below.

In a preferred embodiment of the present invention instruction queues 39 are provided, which effectively the couple fetch and dispatch stages in the pipeline. There are in this embodiment eight instruction queues, one for each stream. In the example of Fig. 3 the instruction queues are shown in a manner to illustrate that each queue may have a different number of instructions ready for transfer to a dispatch stage 41.

Referring again to instruction cache 31 and the two ports to fetch stage 33, it was described above that eight instructions may be fetched to fetch stage 33 by each port. Typically the eight instructions for one port are eight instructions from a single thread for a single stream. For example, the eight instructions fetched by one port in a particular cycle will typically be sequential instructions for a thread associated with one stream.

Determination of the two threads associated with two streams to be accessed in each cycle is made by selection logic 35. Logic 35 monitors a set of fetch program counters 37, which maintain a program counter for each stream, indicating at what address to find the next instruction for that stream. Select logic 35 also monitors the state of each queue in set 39 of instruction queues. Based at least in part on the state of instruction queues 39 select logic 35 determines the two threads from which to fetch instructions in a particular cycle. For example, if the instruction queue in set 39 for a stream is full, the probability of utilizing eight additional instructions into the pipeline from the thread associated with that stream is low. Conversely, if the instruction queue in set 39 for a stream is empty, the probability of utilizing eight additional instructions into the pipeline from the thread associated with that stream is high.

In this embodiment, in each cycle, four instructions are made available to dispatch stage 41 from each instruction queue. In practice dispatch logic is provided for selecting from which queues to dispatch instructions. The dispatch logic has knowledge of many parameters, typically including priorities, instruction dependencies, and the like, and is also aware of the number of instructions in each queue.

As described above, there are in this preferred embodiment ten execution units, which include two memory units 43 and eight arithmetic logic units (ALUs) 45. Thus, in each cycle up to ten instructions may be dispatched to execution units.

In the system depicted by Fig. 3 the unique and novel set of instruction queues 39 provides decoupling of dispatch from fetch in the pipeline. The dispatch stage now has a larger pool of instructions from which to select to dispatch to execution units, and the efficiency of dispatch is improved. That is the number of instructions that may be dispatched per cycle is maximized. This structure and operation allows a large number of streams of a DMS processor to execute instructions continually while permitting the fetch mechanism to fetch from a smaller number of streams in each cycle. Fetching from a smaller number of streams, in this case two, in each cycle is important, because the hardware and logic necessary to provide additional ports into the instruction cache is significant. As an added benefit, unified access to a single cache is provided.

Thus the instruction queue in the preferred embodiment allows fetched instructions to be buffered after fetch and before dispatch. The instruction queue read mechanism allows the head of the queue to be presented to dispatch in each cycle, allowing a variable number of instructions to be dispatched from each stream in each cycle. With the instruction queue, one can take advantage of instruction stream locality, while maximizing the efficiency of the fetch mechanism in the presence of stalls and branches. By providing a fetch mechanism that can support up to eight instructions from two streams, one can keep the instruction queues full while not having to replicate the fetch bandwidth across all streams.

Clustering Streams and/or Instruction Queues

In an alternative embodiment of the present invention a further innovation is made in a multistreaming processor which may or may not have instruction queues associated with streams

Fig. 4 is a block diagram for a pipelined architecture for a multistreaming processor according to another embodiment of the present invention. In the processor illustrated by Fig. 4 there are eight streams, just as in the processor of Fig. 3. There are also eight fetch stages, one for each stream, and a full set of execution units. In this example there are instruction queues shown, one for each stream, but the presence of these queues is not required for the present invention. A salient difference from architecture previously described is that the plurality of streams is grouped into two distinct clusters.

Referring again to Fig. 4, instructions are fetched from instruction cache 47 by two stream clusters 49 and 51, labeled Cluster A and Cluster B. Cluster A comprises four streams, each having a fetch stage 63 and a set of instruction queues 65, one for each stream. The instruction queues operate as described above for the processor of Fig. 3. Cluster A further has a dispatch stage 67 for the four streams in the cluster, which dispatches instructions from queues 65 to a set of functional, or execution units 69.

Cluster B (51) has the same structure as Cluster A, comprising four streams, each with a fetch stage in set 55, each having an instruction queue in set 57, and a dedicated dispatch stage 59 which dispatches instructions from the instruction queues to a set of execution (functional) units 61.

In some embodiments of this unique architecture there are one or more general execution units (GXU) 71, to which instructions may be dispatched by either of dispatch stages 67 or 59. The clusters share a common data cache 53.

Instruction cache 47 still has two ports, as in the previously described embodiment, and there is a select system, much as previously described, for selecting which stream in each cycle in each Cluster will fetch instructions. The select system has access, as before, to FPCs, and monitors the state of each instruction queue in each Cluster. In the present case one stream of four in each Cluster is selected each cycle to fetch eight sequential instructions beginning at the PC address.

Referring again to Fig. 4, there are two dispatch stages, one for each cluster, each of which dispatches instructions from only the queues in its own associated Cluster.

A distinct advantage in clustering streams with use of instruction queues as described and taught herein, is that the overall complexity, hence cost, of implementing two 4x4 clusters is less than implementing the 8x8 array described with the aid of Fig. 3.

The skilled artisan will recognize that there are a number of alterations that might be made in embodiments of the invention described above without departing from the spirit and scope of the invention. For example, the number of instruction queues may vary, the number of ports into the instruction cache may vary, the fetch logic may be implemented in a variety of ways, and the dispatch logic may be implemented in a variety of ways, among other changes that may be made within the spirit and scope of the invention. There also be a different Clustering of streams than that depicted and described as an example herein. For these and other reasons the invention should be afforded the broadest scope, and should be limited only by the claims that follow.